IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 26 January 2024, accepted 12 February 2024, date of publication 21 February 2024, date of current version 4 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3368793

RESEARCH ARTICLE

Research on Task-Offloading Delay in the IoV Based on a Queuing Network

JINGYUN WEI AND XIANGYANG LIANG
School of Computer Science and Technology, Xi'an Technological University, Xi'an 710000, China
Corresponding author: Xiangyang Liang (xiangyangl0913@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 62276146.

ABSTRACT In this paper, we address the challenge of task offloading for mobile edge computing in the Internet of Vehicles (IoV). We employ a closed queuing network model to analyze the optimal server load percentage, focusing on the response time of tasks. We evaluate the delay of task offloading by constructing a closed queuing network model and applying the analysis of this network. Additionally, we analyze the system performance by varying the number of vehicles, the number of edge servers, and the edge server load percentage. The results of the study show that there exists an optimal edge server load percentage during task offloading, and this value makes it possible to compute the minimum average response time spent for a task during offloading while ensuring fairness in offloading delays. Increasing the number of edge servers affects the selection of the optimal load percentage, which in turn reduces the minimum response time. Moreover, as the number of vehicles increases, the average response time of tasks in the system increases accordingly. This paper provides a solution for delay-oriented task offloading in the IoV.

INDEX TERMS Closed queuing network, response time, server load, task offloading.

I. INTRODUCTION

The widespread adoption of such applications as collaborative autonomous driving, intelligent traffic control, and collaborative environment awareness in the IoV has led to an increased demand for extensive data transmission and information exchange among vehicles. Consequently, the amount of vehicle data is growing exponentially and the complexity of computational tasks is increasing. Despite the inherent storage and computational capabilities of vehicles, they are constrained by hardware limitations, rendering them inadequate for handling substantial data volumes and intricate computational tasks. To overcome these limitations, the integration of mobile cloud computing (MCC) with the IoV is proposed for offloading tasks from vehicles to remote cloud centers [1]. Despite the robust storage and computational capabilities of cloud servers, they are usually deployed at remote locations and may not meet the requirements of computational tasks in vehicles. On the one hand, computational tasks in vehicles are closely tied to security and often entail stringent delay requirements. However, longer transmission distances increase the delay [2], which impairs real-time responsiveness, especially for security applications [3]. On the other hand, the battery life of vehicles must also be considered. Due to the longer transmission distance, offloading computational tasks to cloud data centers consumes more energy, which adversely affects the vehicle's range. Therefore, multiaccess edge computing (MEC) has been introduced to the IoV, with mobile edge servers typically deployed in roadside units (RSUs) along the road. Delay and security issues can be effectively mitigated by offloading tasks to edge servers. Efficient task offloading and resource allocation strategies have the potential to greatly enhance system performance.

Traditional task offloading involves offloading tasks exclusively to either the cloud server or the edge server for computation. The goal of offloading is to reduce the total task delay while minimizing vehicle energy consumption. The computing capability of the cloud server significantly exceeds that of vehicles and edge servers, leading to a substantial reduction in task computing time with this offloading method. However, if all the tasks are offloaded to cloud

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

servers, a long transmission distance will result in increased transmission delay, and at the same time, vehicles will experience greater energy consumption. If all the tasks are offloaded to the edge server, the shorter transmission distance can effectively alleviate the transmission delay. However, the computing capability of the edge server is much lower than that of the cloud server, resulting in longer computing times. When numerous vehicle tasks are offloaded to the edge server, the lack of capacity will lead to a long delay in processing the computation tasks. Offloading strategies, whether for edge servers or cloud servers, present significant drawbacks; the primary constraints are due to the limitations of edge servers and the substantial delay introduced by the distance to cloud servers. Therefore, an efficient task-offloading strategy is needed to reduce the delay of computing task-offloading and improve system performance. In this paper, we propose a task scheduling strategy based on queuing networks (TSSQN) for addressing delays in task offloading. The TSSQN consists of a system task scheduling model and a queuing network-based performance metric solver. We further investigate the relationship between vehicle task offloading and system performance from the perspective of computational task offloading delay. The TSSQN can analyze the optimal offloading strategy when faced with different road situations, such as varying vehicle densities and RSU configurations.

In this article, each task of a vehicle is independent. Therefore, closed queuing networks are an ideal research method for addressing the challenge of task offloading in the IoV [4], [9], [15]. This paper makes several key contributions.

- We propose a large-scale solution for the delay of computational offloading in the IoV that is applicable across various road scenarios.
- We introduce a multiclass closed queuing network to model the task offloading problem in edge computing for the IoV. Additionally, we analyze the system performance based on the task response time, a key performance metric for task offloading in the IoV.
- We find that the server load has a significant impact on the task response time. By analyzing the queuing network, we find that there is always an optimal load that minimizes the response time in all cases. Moreover, we observe that the load percentage of the edge server that achieves the minimum response time closely aligns with the delay fairness among the different offloading methods.
- We discover that the performance of the queuing network is influenced by several factors, including the number of vehicles and the number of edge servers. Adjusting these parameters allows us to meet the delay requirements of vehicle tasks.

The paper is organized as follows. Section II provides an overview of related work in this area. In Section III, we model the task offloading problem for edge computing in the IoV. Section IV derives the model and provides a solution for the task response time. In Section V, we calculate the system performance and analyze it using queuing networks. Finally, Section VI summarizes the paper.

II. RELATED WORK

Edge computing in the context of the IoV has been extensively studied by many scholars. Notably, Garg et al. [5] proposed utilizing edge computing platforms as RSUs for vehicular communication. Moreover, they designed an intelligent security framework for VANETs based on edge nodes and 5G to improve communication and computational capabilities. In reference [6], an SDN-IoV network architecture that includes an end layer, an edge layer, and a cloud layer was proposed by combining mobile edge computing technology and software-defined networking technology. The controller placement problem was studied in terms of delay and controller load balancing. In reference [7], the computational efficiency of a task was used as a system performance metric; this was defined as the ratio of computational bits to the energy consumption of a task. Additionally, the authors proposed the MACTER algorithm for offloading decisions. In reference [8], vehicle task offloading was considered a multi-objective constrained optimization problem, and a non-dominated sorting genetic strategy (NSGS) was proposed to solve the problem; this approach reduces the delay and energy consumption of task offloading. However, relying only on edge computing or cloud computing to perform tasks is limited either by the lack of computational capabilities or by excessive transmission delay. To solve these problems, Liu et al. [9] synthesized the delay, cost, and energy consumption factors proposed a cloud-edge-end collaborative computing task offloading model based on a queuing network, and examined the effects of optimized offloading probability and transmit power on delay, cost, and energy consumption. In this paper, cloud-edge collaboration was used to offload some of the computing tasks to edge servers and some of them to cloud servers to reduce the delay of computing tasks. Additionally, in reference [10], the utilization of the vehicular cloud for managing computationally intensive tasks was suggested as a strategy to address the issue of the limited computational resources of individual vehicles. The proposed solution involved breaking down the vehicle's task into several interconnected subtasks and distributing these subtasks across multiple vehicles in a decentralized manner. Additionally, the PR-VC was introduced to enhance the conditional mean time to failure, mitigating challenges associated with resource fluctuations resulting from the movement of vehicles. Vehicle movement at high speeds also causes the topology to change rapidly, adding unnecessary delays. Tariq et al. [11] addressed the challenges posed by vehicle movement by proposing a combination of an NDN and an SDN called NDSDoV. NDSDoV uses edge controllers to maintain and manage real-time vehicular topology. The SDN controllers use the information received from the edge controllers to maintain their tables, including a global information table and routing information table and finally forward it according to the forwarding mechanism.

Queuing theory has a unique advantage in studying system performance because it reduces the complexity of the problem and facilitates research. In reference [12], spatial-temporal event processing (STEP) was used to process spatiotemporal data streams for the IoV, and a queuing network model was utilized to evaluate the response time of the STEP temporal stream processing system. In reference [13], the authors modeled the cloud computing system as an M/G/1 queue and proposed a task scheduling algorithm based on similar tasks to reduce energy consumption. In reference [14], a scheduling model was established for public cloud platforms to validate an effective platform scheduling algorithm, and C1oudSim was used to conduct simulation experiments on different scheduling strategies with queuing models; these models consider virtual machines as service organizations and dynamically adjust their number to adapt to scale changes in system applications. Luo et al. [4] modeled the process of multiuser requests for mobile edge computing systems as an M/M/1 queue, the problem of minimizing the total response time of all tasks was examined, and for this optimization problem, a greedy algorithm was proposed, which tends to assign tasks to the server with the minimum response time. The simulation results showed that the mean response time of the tasks was reduced by 20%∼30% for the proposed greedy algorithm compared to that of the randomized algorithm. In reference [15], the authors modeled fog devices as M/M/1 queues and cloud systems as M/M/C queues. The fog-cloud workload distribution problem was solved by using the NSGA-II algorithm, which aims to reduce energy consumption and delay.

Queueing networks serve as powerful tools for probing the essence of problems, playing a role not only in algorithmic improvements but also in fostering a comprehensive understanding of the nature of problems, performance analysis, and system optimization. Their applications extend widely across various domains. In reference [16], the authors analyzed the end-to-end average delay and maximum achievable single-node throughput of multi-hop wireless ad hoc networks based on closed queuing networks. Each node was modeled as a G/G/1 queue, and expressions for the average delay, node throughput, and node average service time were derived via diffusion approximation. The simulation results were compared with the analytical results, and they aligned closely. In reference [17], Wireless Sensor Networks (WSNs) were modeled as open G1/G/1/N queuing networks, and analytical formulas for performance metrics such as the end-to-end average delay, packet loss probability, throughput, and average hop count were analyzed. This study provided numerical examples for the performance evaluation of intermittent network models. In reference [18], they conducted a coarse-grained comparative analysis of simple fine-grained multicore processors and complex synchronous multithreaded multicore processors for server applications with high parallel requests. Using queuing networks, the throughput performance of processors was derived. This approach yields different core counts, thread counts, and various workloads. In reference [19], a closed queuing network model with multiple servers was introduced to simulate data flow in a multi-threaded architecture. This paper discussed performance metrics related to queuing models, such as queue length, response time, throughput, and utilization, providing results for the multiserver queuing model.

In existing research, there has been a predominant focus on resource management and scheduling processes, with insufficient attention given to the challenge of balancing server load ratios to minimize delays during task scheduling. Additionally, the issue of delay fairness for tasks has been overlooked. To address these gaps, this paper places a primary emphasis on computing task scheduling in a cloud-edge environment. We propose a task scheduling strategy based on queuing networks (TSSQN) specifically tailored to the cloud-edge environment, aiming to analyze optimal solutions for different scenarios to optimize task offloading strategies.

In this paper, each task of a vehicle is independent and after task offloading, the results need to be transmitted back to the vehicle. Therefore, a closed queuing network is an ideal research approach when addressing the challenge of task offloading in the IoV. Using the queuing network to model the task offloading problem for edge computing in the Internet of Vehicles, this paper uses delay as a system performance metric and optimizes the system performance by optimizing the server load.

III. MODELING

The behavior of task offloading using edge computing architectures in the IoV can be modeled as a closed queuing network. There are multiple job classes circulating in the queueing network, and a job class is a collection of jobs with the same behavior, as reflected by the fact that these jobs have the same routing probability and service time distribution. For example, for a queuing network with K vehicles, it is necessary to define K classes of jobs, where each job class corresponds to the jobs of a certain vehicle. This is because when the server returns the results from processing the computational task, the jobs need to be returned to the original vehicle node; therefore, the jobs of each vehicle are modeled as the same class. We model a BCMP queuing network which is a hybrid queuing network that has great advantages in solving complex problems.

As shown in **Figure 1**, we consider a BCMP closed queuing network model. The model includes K vehicles, a transmission link, an edge layer, and a cloud server. This results in $K+3$ nodes, with the edge layer incorporating m edge servers. We argue that under general road conditions, the density of vehicles tends to remain constant over time. For modeling purposes, we consider each computational task of a vehicle as one job in the system. This abstraction allows us to study computational tasks independently, increasing relevant parameters (e.g., service rate) when a vehicle carries multiple homogeneous jobs. Therefore, each vehicle entering a fixed roadway is assumed to have only one job. The number of tasks in the system is determined by the number of vehicles,

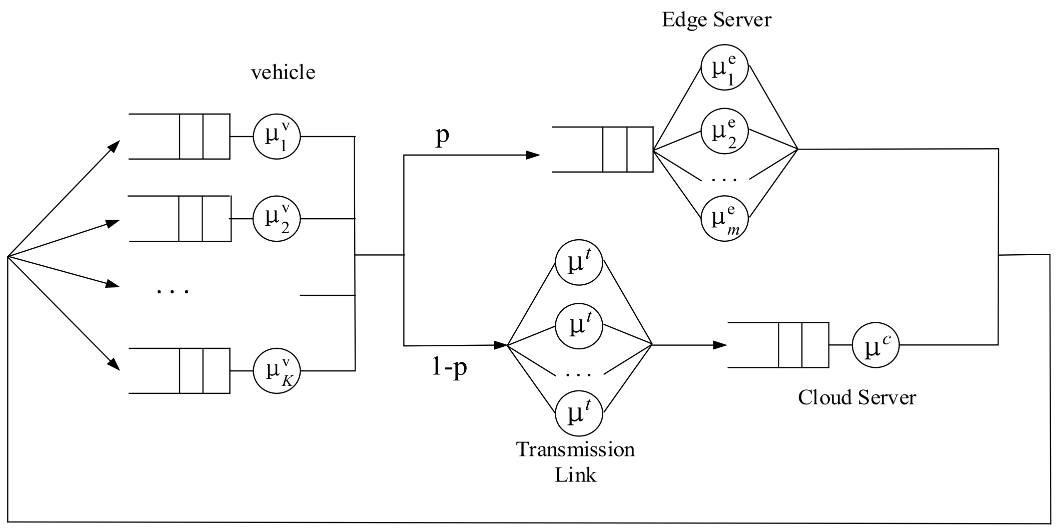

FIGURE 1. System model.

resulting in K jobs cycling through the queuing network. Throughout the entire task offloading process, tasks do not become other vehicles tasks, meaning that there is no class swapping in the queuing network. Consequently, we have K classes of jobs. The computational tasks are offloaded by the vehicles, and each vehicle node is independent, so each vehicle can be viewed as an M/M/1 queue; the service time of the vehicle nodes when offloading the tasks is negatively exponentially distributed, and the service rate is μ_i^v, where i =1, …, K. In vehicle task offloading with cloud-edge collaboration, there are two choices for computing tasks when offloading. One is to offload the task to the edge server, which is deployed in the RSU at a close transmission distance; thus, the transmission time of task offloading can be ignored, and only the edge layer needs to be considered in modeling. The second is the choice of offloading to the cloud server, which needs to be considered when modeling. In addition, by incorporating the location of the cloud server the transmission time of task offloading must be considered. When a job is offloaded from the ith vehicle, the probability of being routed to an edge layer node is p and the probability of being routed to the cloud server node is *1-p*. When the job exits from the ith vehicle node, we assume that the probability of the job entering the edge layer is p, the probability of entering the transmission link between the vehicle and the cloud server is *1-p*, and the probability of the job routing to the cloud server after leaving the transmission link is 1. Jobs are routed back to the original vehicle after receiving service. In the system, jobs follow the first-come-first-served (FCFS) rule.

In our model, we represent the edge layer and cloud servers as an M/M/m queue and an M/M/1 queue, respectively. For a fixed road section, edge servers are often uniformly distributed along the road, let the number of edge servers on a road section be denoted as "m." The entire edge layer can then be effectively modeled as an M/M/m queue, where "m" represents the number of edge servers. Tasks are queued after arriving at the edge layer, and only when idle edge servers are available are the tasks in the queue served according to certain rules (e.g., FCFS). The service rate of the edge servers is denoted as μ_i^e, where i =1, …, m. The service rate represents the number of jobs that can be completed per unit of time, i.e., the inverse of the time that the job receives the service at the node is the service rate of the node. For example, the reciprocal of $m\mu_i^e$ is the mean time for the server to process the computed jobs. It is determined by the magnitude of the job and the processing capabilities of the server. A cloud server as a cluster can be considered a large server from a macro point of view, modeling the cloud server as an M/M/1 queue. Its performance is much stronger than that of edge servers, and this can be represented by a large service rate, denoted as μ^c. The service rate of the cloud server is much greater than that of the edge servers; i.e., $\mu^c \gg \mu^e$. When μ^c is large enough, a new task arriving at the cloud can be served very quickly. The transmission process of offloading tasks to the cloud server requires the task to be sent to the RSU first, as the RSU acts as a relay node that forwards the task to the cloud server. The process of transmitting tasks from vehicles to cloud servers can be holistically modeled as an M/G/∞ queue. Notably, task offloading to the cloud server requires long-distance transmission, so we need to set an appropriate value for μ^t in the queuing network.

For the sake of analysis, we assume that the tasks are homogeneous for all vehicles. Specifically, for vehicles, the transmission link, edge servers, and the cloud server, the service rates for computational tasks from different vehicles are consistent and obey negative exponential distributions. To investigate the optimal offloading policies in different scenarios, we compare the mean response times of 20, 30, 40, and 50 vehicles and 2, 3, 4, and 5 edge servers.

IV. MODEL SOLVING

According to the analysis in Section III, this system is a closed queuing network containing K+3 nodes. The queuing network contains two classes. One is an M/M/m queue

(M/M/1 queue when $m = 1$) with the first-come first-served (FCFS) queuing rule, and the other is an M/G/∞ queue with the Infinite Server (IS) queuing rule. The vehicle nodes and server nodes with the FCFS queuing rule have the same service rate for each type, and this system is a BCMP queuing network with a solution in product form.

The BCMP network can contain multiple job classes and multiple types of queuing nodes, and the whole queuing network is no longer limited to open or closed. It can solve open queuing networks, closed queuing networks, and even mixed queuing networks. In BCMP, the normalization constant, denoted as $G\left(\overrightarrow{K}\right)$, is the sum of the products of the functions of the state probabilities of each node in all possible states, and the marginal probability can be derived from the normalization constant $G\left(\overrightarrow{K}\right)$. The marginal probability describes the probability of each possible state when the system is in a steady state. System performance metrics such as throughput, queue length, and response time can be derived from the marginal probability and normalization constant. Here, we can accurately derive the response time of the closed queueing network model using BCMP theory. Next, we provide the solution for the response time of the model in this paper.

Let the vehicles be the 1st, 2nd, …, Kth nodes, the edge layer be the $K+1$st node, the transmission link be the $K+2$nd node, and the cloud server be the $K+3$rd node. The queuing network is in a steady state and the arrival rate of a job of the rth class on the ith node must satisfy the equilibrium equation so that the access rate of the ith vehicle, transmission link, or server is

$$e_{ir} = \sum_{j=1}^{K+3}\sum_{s=1}^{K} e_{js}p_{js,ir} \tag{1}$$

Assuming that there are K vehicle nodes in the system, only one class of task is sent or received on each vehicle node, and no class switching occurs when the task is transmitted or computed in the system, then for the vehicle nodes, $i = 1, \ldots, K$, $r = 1, \ldots, K$, we have

$$e_{ir} = \begin{cases} 1, & i = r \\ 0, & i \neq r \end{cases} \tag{2}$$

In the absence of class switching, when a job is offloaded from the ith vehicle, the probability of being routed to an edge layer node is denoted as $p_{js,(K+1)s} = p$, and the probability of being routed to a transmission link is denoted as $p_{js,(K+2)s} = 1 - p$. Therefore, we have

$$p_{js,is} = \begin{cases} p, & 1 \le j \le K, i = K+1 \\ 1-p, & 1 \le j \le K, i = K+2 \end{cases} \tag{3}$$

The job will be routed to the cloud server with a probability of 1 after leaving the transport link, i.e., the routing probability $p_{(K+2)s,(K+3)s} = 1$. The job is routed to the vehicle with a probability of 1 from the edge layer and the cloud server. Therefore, we have

$$p_{js,is} = 1,\ j = K+1,\ K+3,\ i = 1,\ \ldots,\ K \tag{4}$$

So, the arrival rate of individual nodes in the queuing network can be expressed as

$$e_{ir} = \begin{cases} p, & i = K+1 \\ 1-p, & i = K+2 \; or \; i = K+3 \\ 1, & i = r \\ 0, & otherwise \end{cases} \tag{5}$$

$G(\overrightarrow{K})$ is a normalization constant, and $\overrightarrow{K} = (K_1, \ldots, K_K)$, denotes the class of vehicle tasks in the system. $\overrightarrow{S_i} = (k_{i1}, \ldots, k_{iK})$, k_{ir} is the number of vehicle tasks of rth class on the ith node, and $\overrightarrow{S_i}$ represents the number of vehicle tasks of each class on the ith node, then $\sum_{i=1}^{K+3}\overrightarrow{S_i} = \overrightarrow{K}$. $G(\overrightarrow{K})$ is defined as

$$G\left(\overrightarrow{K}\right) = \sum_{\sum_{i=1}^{K+3}\overrightarrow{S_i} = \overrightarrow{K}} F_1\left(\overrightarrow{S_1}\right)\cdot F_2\left(\overrightarrow{S_2}\right)\cdot \ldots \cdot F_{K+3}\left(\overrightarrow{S_{K+3}}\right) \tag{6}$$

$F_i\left(\overrightarrow{S_i}\right)$ is a function of the state probability with respect to either the vehicle nodes or the edge server node, and the vehicle nodes and the server node are M/M/m queues (vehicle nodes with $m = 1$) with

$$F_i\left(\overrightarrow{S_i}\right) = k_i!\frac{1}{\beta_i\left(k_i\right)}\cdot q\left(\frac{1}{\mu_i}\right)^{k_i}\cdot q\prod_{r=1}^{K}\frac{1}{k_{ir}!}e_{ir}^{k_{ir}}, \quad i = 1, \ldots, K, K+1, K+3 \tag{7}$$

Transmission nodes are M/G/∞ queued with

$$F_i\left(\overrightarrow{S_i}\right) = \prod_{r=1}^{K}\frac{1}{k_{ir}!}\cdot q\left(\frac{e_{ir}}{\mu_{ir}}\right)^{k_{ir}}, \quad i = K+2 \tag{8}$$

Function $\beta_i\left(k_i\right)$ defined as

$$\beta_i\left(k_i\right) = \begin{cases} k_i!, & k_i \le m_i \\ m_i!\cdot m_i^{k_i - m_i}, & k_i \ge m_i \\ 1, & m_i = 1 \end{cases} \tag{9}$$

Both the vehicle and the cloud server are M/M/1 queues, i.e., $m = 1$, and thus there are

$$F_i\left(\overrightarrow{S_i}\right) = \begin{cases} k_i!\cdot\left(\dfrac{1}{\mu_i}\right)^{k_i}, & i = 1, \ldots, K \\ k_i!\dfrac{1}{\beta_i\left(k_i\right)}\cdot\left(\dfrac{1}{\mu_i}\right)^{k_i}\cdot\prod_{r=1}^{K}p^{k_{ir}}, & i = K+1 \\ \prod_{r=1}^{K}\left(\dfrac{e_{ir}}{\mu_{ir}}\right)^{k_{ir}}, & i = K+2 \\ k_i!\cdot\left(\dfrac{1}{\mu_i}\right)^{k_i}\cdot\prod_{r=1}^{K}(1-p)^{k_{ir}}, & i = K+3 \end{cases} \tag{10}$$

Definition $\left(\vec{K}-1_r\right)=(K_1,\ldots,K_r-1,\ldots,K_K)$, the throughput is

$$\lambda_r=\frac{G\left(\vec{K}-1_r\right)}{G\left(\vec{K}\right)} \tag{11}$$

Then the vehicle node and the server node have throughput for the task on the rth vehicle

$$\lambda_{ir}=e_{ir}\lambda_r=e_{ir}\frac{G\left(\vec{K}-1_r\right)}{G\left(\vec{K}\right)} \tag{12}$$

For M/M/1 vehicle nodes, we have

$$\lambda_{ir}=e_{ir}\frac{G\left(\vec{K}-1_r\right)}{G\left(\vec{K}\right)}=\frac{G\left(\vec{K}-1_r\right)}{G\left(\vec{K}\right)},\quad i=r \tag{13}$$

For edge servers, we have

$$\lambda_{(K+1)r}=e_{ir}\frac{G\left(\vec{K}-1_r\right)}{G\left(\vec{K}\right)}=p\frac{G\left(\vec{K}-1_r\right)}{G\left(\vec{K}\right)} \tag{14}$$

For cloud servers, we have

$$\lambda_{(K+3)r}=e_{ir}\frac{G\left(\vec{K}-1_r\right)}{G\left(\vec{K}\right)}=(1-p)\frac{G\left(\vec{K}-1_r\right)}{G\left(\vec{K}\right)} \tag{15}$$

The steady-state probability $\pi\left(\vec{S_1},\ldots,\overrightarrow{S_{K+3}}\right)$ can be described as

$$\pi\left(\vec{S_1},\ldots,\overrightarrow{S_{K+3}}\right)=\frac{1}{G\left(\vec{K}\right)}\prod_{i=1}^{K+3}F_i\left(\vec{S_i}\right) \tag{16}$$

The marginal probability $\pi_i\left(\vec{K}\right)$ is denoted as

$$\pi_i\left(\vec{K}\right)=\sum_{\sum_{j=1}^{K+3}\vec{S_j}=\vec{K}\ and\ \vec{S_i}=\vec{K}}\pi\left(\vec{S_1},\ldots,\overrightarrow{S_{K+3}}\right) \tag{17}$$

Both the mean number of tasks and node utilization of the system are related to the marginal probability $\pi_i\left(\vec{K}\right)$, and the mean number of tasks of class r on the ith node $\overline{K_{ir}}$ is

$$\overline{K_{ir}}=\sum_{all\ \vec{K}\ with\ k_r>0}k_r\pi_i\left(\vec{K}\right) \tag{18}$$

where $i=1,\ldots,K+3$ and $r=1,\ldots,K$. According to Little's theorem [20], the response time $\overline{T_{ir}}$ of a task of rth class at the ith node is

$$\overline{T_{ir}}=\begin{cases}\dfrac{\overline{K_{ir}}}{\lambda_{ir}}, & i=1,\ldots,K,K+1,K+3\\ \dfrac{1}{\mu_{ir}}, & i=K+2\end{cases} \tag{19}$$

TABLE 1. Parameters values.

Parameter	Value
K	20, 30, …, 50
K_r	1
$(\mu_{11},\mu_{22},\ldots,\mu_{KK})$	(100, 100, …, 100)
$(\mu_{(K+1)r},\mu_{(K+2)r},\mu_{(K+3)r})$	(0.1,0.5,3.3)
m	2,3,4,5
p	0.1, 0.2, …, 0.9

consequently

$$\overline{T_{ir}}=\begin{cases}\dfrac{\overline{K_{ir}}}{\frac{G\left(\vec{K}-1_r\right)}{G\left(\vec{K}\right)}}, & i=r\\ \dfrac{\overline{K_{ir}}}{p\frac{G\left(\vec{K}-1_r\right)}{G\left(\vec{K}\right)}}, & i=K+1\\ \dfrac{1}{\mu_{ir}}, & i=K+2\\ \dfrac{\overline{K_{ir}}}{(1-p)\frac{G\left(\vec{K}-1_r\right)}{G\left(\vec{K}\right)}}, & i=K+3\end{cases} \tag{20}$$

In this system, the mean response time of task $\overline{T_r}$, includes the time taken by the task to send, transmit, compute and send back the result from the vehicle. However, when the computational task is processed and the amount of data in the computational result is much less than that of the computational task itself, the time spent transmitting to the vehicle is negligible; thus, the total time is

$$\overline{T_r}=\overline{T_{rr}}+p\overline{T_{(K+1)r}}+(1-p)\left(\overline{T_{(K+2)r}}+\overline{T_{(K+3)r}}\right) \tag{21}$$

where $r=1,\ldots,K$.

V. RESULTS AND ANALYSIS

In this section, we provide numerical results for the performance metrics of the system network model for different values of m and K. Since the focus of our paper is to analyze the average end-to-end delay of task offloading in the IoV due to different offloading strategies, we vary the offloading load percentage p. Based on the analysis of the system model in Section III, this section provides performance results for multiple types of a closed queuing network. The parameter values are detailed in **Table 1**.

In the system, the vehicle offloads tasks at a high rate, allowing the service rate of the vehicle to be set significantly higher than the service rates of the edge servers and the cloud server, for instance, at 100. In this paper, we adopt C-V2X for communication, assuming that the channel bandwidth of the transmission link is 10 MHz [21]. If the size of the offloaded task is 10 MB, considering the actual situation of the transmission link, such as the signal-to-noise ratio [22], by calculation, we can set the transmission link to 3.3 computing tasks per unit time. The parameter settings of the edge

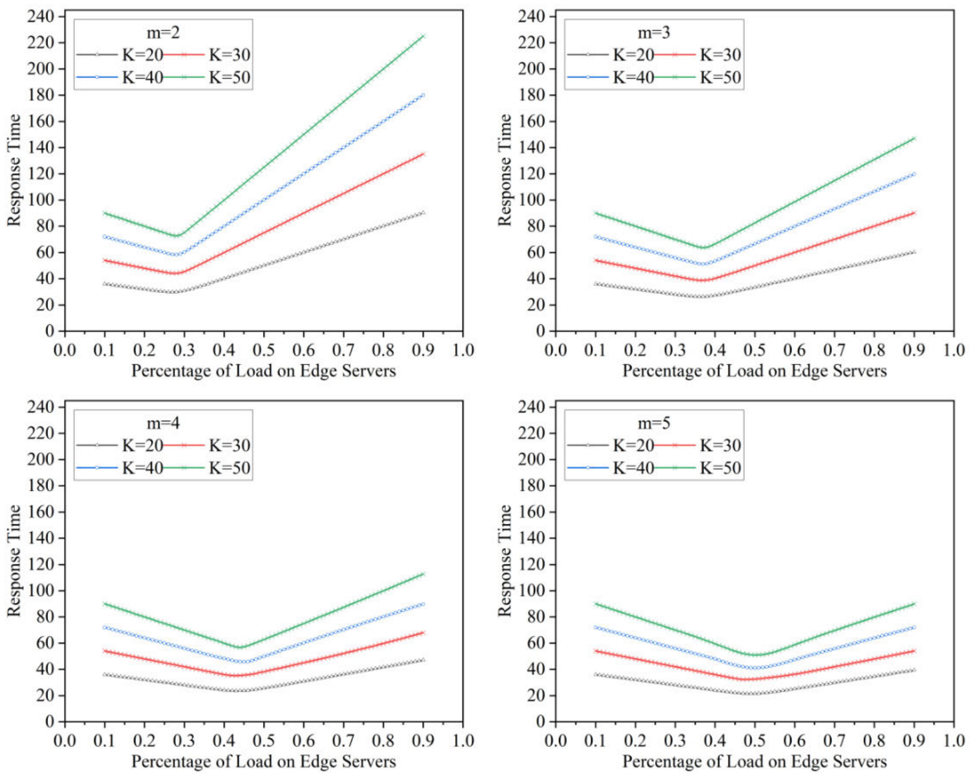

FIGURE 2. Response times of computing tasks with different edge server load percentages.

server refer to [22], and we calculate that 0.1 computing tasks should be served per unit time. We set the capacity of the cloud server to 5 times that of the edge server; i.e., 0.5 computing tasks are served per unit time. The different service rates are an indication of the computational capabilities of the edge server and the cloud server. The number of vehicles determines the number of job classes in the closed queuing network and each vehicle carries only one job when it enters the roadway. We explore the impact of server load on the task response time by setting different numbers of vehicles and different numbers of edge servers.

We determine the computation task response time versus edge load for different numbers of edge servers and consider the computation task response time for different edge server load scenarios, as shown in **Figure 2**, plotted sequentially for $m = 2$, 3, 4 and 5 as the number of edge servers.

First, by comparing the four plots in **Figure 2**, we note that for the cloud-edge cooperative task offloading system, deploying more edge servers in a fixed-length road segment can effectively reduce the response time of tasks. This is because if the capacity of a single edge server to handle a task is μ_i^e, then the capacity of an edge layer with m edge servers will be $m\mu_i^e$, which increases as m increases.

Second, by analyzing any curve in **Figure 2**, it can be found that when the vehicle density is constant, the task response time first decreases and then increases with the increase in the load percentage of edge servers. When the load percentage of edge servers is very small, the edge server can afford to

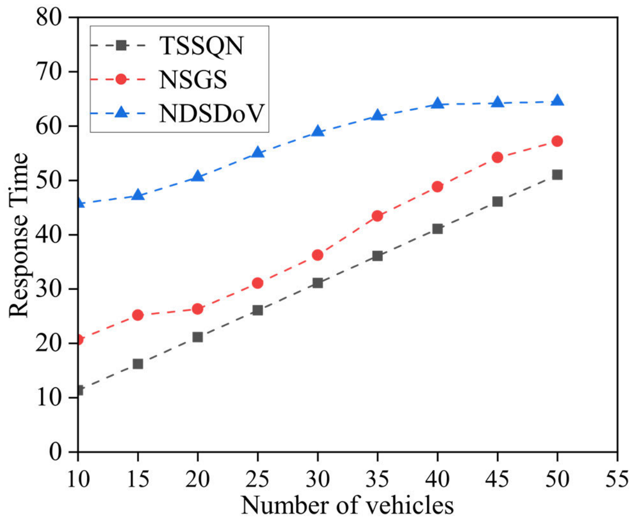

FIGURE 3. Response times for different algorithms.

offload tasks, so the time spent by the edge server to process the tasks is very short. However, for the cloud server, the remaining tasks are offloaded to it, and the time needed to process the computing tasks increases due to the overburdening of the cloud server. Moreover, the transmission delay for task offloading is longer. Therefore, the response time to the tasks initially decreases. In this scenario, within the task load that the edge server can afford, offloading more computing

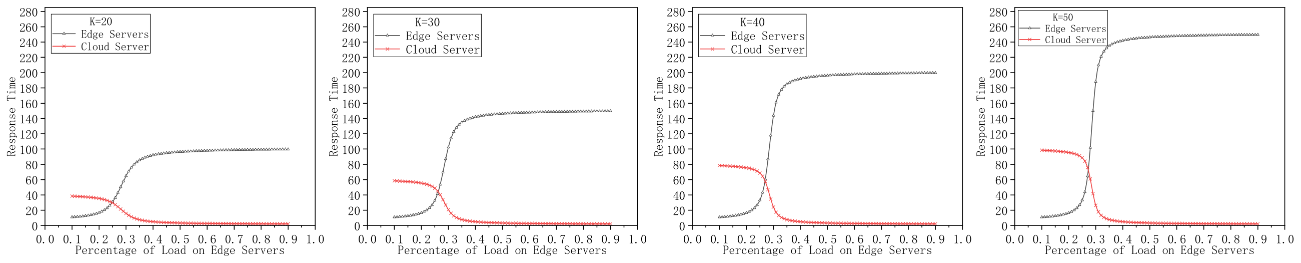

FIGURE 4. Response times of the edge and cloud servers for $m = 2$.

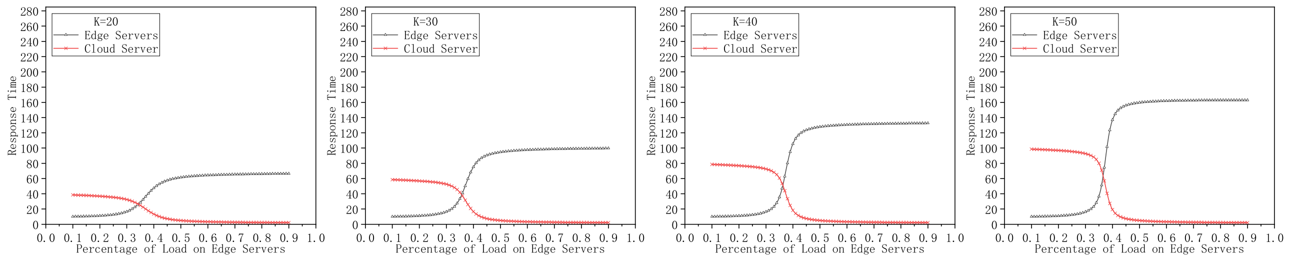

FIGURE 5. Response times of the edge and cloud servers for $m = 3$.

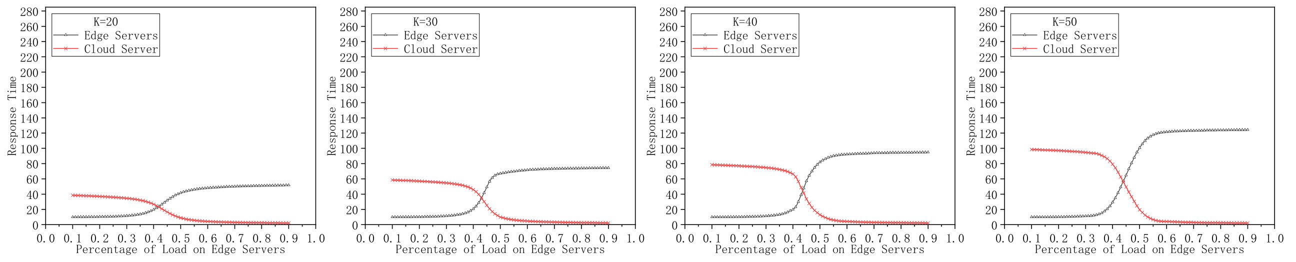

FIGURE 6. Response times of the edge and cloud servers for $m = 4$.

tasks to edge servers can effectively reduce the mean response time of the tasks. However, when the load on the edge server exceeds the maximum, offloading more computing tasks to the edge server will increase the processing time of the task significantly. Although the response time of the cloud server becomes shorter as the load becomes lighter, the response time of the edge tier increases much more than the response time of the cloud server decreases because it processes tasks that exceed the optimal load percentage. However, offloading more computing tasks to edge servers will increase the task response time. There exists an optimal load percentage for the edge servers, and offloading computing tasks rationally according to the optimal load percentage can minimize the mean response time of the tasks.

Third, when the number of edge servers is known, the greater the number of vehicles is, the longer the response time of the task. This is because at larger vehicle densities, the more loaded side of either the edge server or the cloud server needs to handle more computing tasks; thus, the more vehicles there are, the longer the response time of the system for the same percentage of load on the edge server.

We compare the task latency for different numbers of vehicles with different offloading strategies for 5 edge servers. For instance, when the number of vehicles is 50, TSSQN chooses to unload 50% of the tasks to edge servers and the remaining tasks to cloud servers. The NSGA strategy involves dividing vehicle computing tasks into smaller unloading tasks and unloading them to local and edge servers. NDSDoV finds the shortest and optimal path between the centralized EC and SDN. **Figure 3** shows that as the number of vehicles increases, the average latency of all three algorithms increases, with the proposed TSSQN achieving the minimum average latency. This is primarily attributed to the simultaneous consideration of both edge servers and cloud servers in task unloading, along with the identification of the optimal unloading percentage, which significantly reduces the task processing time. NSGA and NDSDoV rely solely on edge computing for task unloading, and the limited capacity of edge servers leads to increased latency when all tasks are unloaded to edge servers, making the latency higher than that of the cloud-edge collaborative TSSQN. Additionally, TSSQN comprehensively considers environmental factors, such as the computing capabilities of edge servers and cloud servers and channel transmission rates. With such considerations, precise offloading strategies can be derived for more complex network

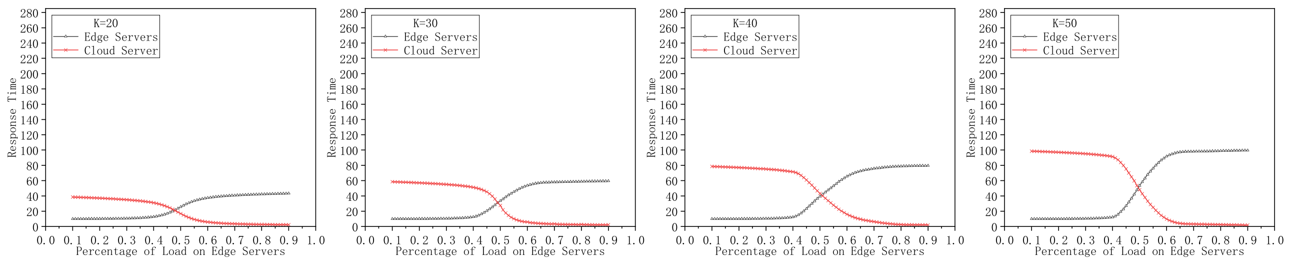

FIGURE 7. Response times of the edge and cloud servers for *m* = 5.

environments, thereby enhancing the latency performance of TSSQN.

Although TSSQN provides different unloading strategies for varying numbers of vehicles, **Figure 3** indicates that as the number of vehicles increases, the latency linearly increases. This is because the optimal unloading percentage provided by the TSSQN results in an increased actual number of tasks unloaded to the edge or cloud as the number of vehicles grows. The increased server load leads to longer processing times. This increasing trend also validates the conclusions drawn in Section IV, indicating that the latency performance is influenced by the computing capacity of the edge layer, the offloading percentage, and the number of vehicles.

There is another issue to consider when offloading computing tasks: whether the delays of the computing tasks offloaded to edge servers and those offloaded to cloud servers are equal when offloading tasks with a load percentage of the minimum response time. If choosing different offloading methods results in different latencies, it is unfair to compute tasks that require more time for task offloading. Therefore, we need to consider delay fairness when performing computing task offloading.

To examine delay fairness, we perform some research. **Figures 4-7** show the variations in the response times of edge servers and cloud servers corresponding to different edge server load percentages for m = 2, m=3, m = 4, and m = 5. In **Figures 4-7**, the X-axis of the intersection represents the percentage of edge load that results in equal latency, while the y-axis indicates the magnitude of task latency under fairness conditions.

First, we find that the edge server load percentage for achieving delay fairness is independent of the number of vehicles and is more strongly affected by the capability of the edge layer. The more capable the edge layer is, the more tasks can be offloaded to the edge servers; thus, the load percentage becomes larger. In addition, we find that when the edge server load percentage increases to a certain value (influenced by m), the response time of the edge servers increases rapidly, while the response time of the cloud servers decreases to a lesser extent than the response time of the edge servers increases. This explains the increasing trend of the total response time in **Figure 2**. Second, comparing **Figure 4** and **Figure 7**, we find that the response time of the edge servers can be reduced by enhancing the capability of the edge layer. Therefore, by estimating the density of vehicles on the road, we can determine how many edge servers need to be deployed to meet the service demand of the vehicles. Finally, in **Table 2**, we compare the edge server load percentage that achieves delay fairness in **Figures 4-7** with the edge server load percentage that achieves the minimum response time in **Figure 2**. Interestingly, we observe that the difference between the load percentages required to achieve minimum response time and delay fairness is consistently less than 2%, and examining the mathematical factors behind this phenomenon is a research goal for our future work. In conclusion, the data are sufficient to show that our strategy of offloading tasks with optimal load percentages is effective at both minimizing the response time and ensuring delay fairness.

TABLE 2. Edge server load percentages for minimum response time and delay fairness.

		Number of vehicles			
		K=20	K=30	K=40	K=50
m=2	Min response time	0.27	0.28	0.28	0.28
	Delay fairness	0.25	0.26	0.27	0.27
	Difference	0.02	0.02	0.01	0.01
m=3	Min response time	0.36	0.37	0.37	0.37
	Delay fairness	0.34	0.36	0.36	0.36
	Difference	0.02	0.01	0.01	0.01
m=4	Min response time	0.43	0.43	0.45	0.44
	Delay fairness	0.42	0.43	0.44	0.44
	Difference	0.01	0	0.01	0
m=5	Min response time	0.49	0.49	0.5	0.5
	Delay fairness	0.48	0.49	0.5	0.5
	Difference	0.01	0	0	0

VI. CONCLUSION

In this paper, we investigate task offloading strategies concerning response time performance in the IoV. By modeling a closed queuing network for the task offloading scenario of edge computing in the IoV, the performance of the queuing network is found to be related to the marginal probability, which in turn is related to the number of tasks in the system, the number of servers, the service rate, and the server load.

We find that for a specific road section, if the number of edge servers is constant, an increase in the number of vehicles will lead to an increase in the mean response time of a single task. If the number of vehicles is constant, the response time can be reduced by adding edge servers. For any number of vehicles and edge servers, there is always an optimal offloading strategy that minimizes the response time of a task while ensuring delay fairness.

In future work, we will further investigate the relationship between the minimum response time and delay fairness. Moreover, we plan to consider the construction cost of servers to reduce the cost as much as possible while satisfying the delay requirements of computing tasks, and the trade-off of the two factors to find the optimal offloading strategy for computing tasks from multiple perspectives.

REFERENCES

[1] D. Van Le and C.-K. Tham, "Quality of service aware computation offloading in an ad-hoc mobile cloud," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8890–8904, Sep. 2018.

[2] Q. Zhang, Y. Wang, X. Zhang, L. Liu, X. Wu, W. Shi, and H. Zhong, "OpenVDAP: An open vehicular data analytics platform for CAVs," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Vienna, Austria, Jul. 2018, pp. 1310–1320.

[3] W. Huang, Y. Huang, S. He, and L. Yang, "Cloud and edge multicast beamforming for cache-enabled ultra-dense networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3481–3485, Mar. 2020.

[4] L. Yuchong, W. Jigang, W. Yalan, and C. Long, "Task scheduling in mobile edge computing with stochastic requests and M/M/1 servers," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Communications; IEEE 17th Int. Conf. Smart City; IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Zhangjiajie, China, Aug. 2019, pp. 2379–2382.

[5] S. Garg, A. Singh, K. Kaur, G. S. Aujla, S. Batra, N. Kumar, and M. S. Obaidat, "Edge computing-based security framework for big data analytics in VANETs," *IEEE Netw.*, vol. 33, no. 2, pp. 72–81, Mar. 2019.

[6] B. Li, X. Deng, and Y. Deng, "Mobile-edge computing-based delay minimization controller placement in SDN-IoV," *Comput. Netw.*, vol. 193, Jul. 2021, Art. no. 108049.

[7] S. Raza, S. Wang, M. Ahmed, M. R. Anwar, M. A. Mirza, and W. U. Khan, "Task offloading and resource allocation for IoV using 5G NR-V2X communication," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10397–10410, Jul. 2022.

[8] J. Zhang, M.-J. Piao, D.-G. Zhang, T. Zhang, and W.-M. Dong, "An approach of multi-objective computing task offloading scheduling based NSGS for IOV in 5G," *Cluster Comput.*, vol. 25, no. 6, pp. 4203–4219, Dec. 2022.

[9] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.

[10] P. Abdisarabshali, M. Liwang, A. Rajabzadeh, M. Ahmadi, and S. Hosseinalipour, "Decomposition theory meets reliability analysis: Processing of computation-intensive dependent tasks over vehicular clouds with dynamic resources," *IEEE/ACM Trans. Netw.*, vol. 32, no. 1, pp. 475–490, Feb. 2023, doi: 10.1109/TNET.2023.3286709.

[11] A. Tariq, I. Ud Din, R. Asif Rehman, and B.-S. Kim, "An intelligent forwarding strategy in SDN-enabled named-data IoV," *Comput., Mater. Continua*, vol. 69, no. 3, pp. 2949–2966, 2021.

[12] H. Li, X. Wu, and Y. Wang, "Dynamic performance analysis of STEP system in Internet of Vehicles based on queuing theory," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–13, Apr. 2022.

[13] C. Cheng, J. Li, and Y. Wang, "An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing," *Tsinghua Sci. Technol.*, vol. 20, no. 1, pp. 28–39, Feb. 2015.

[14] J. Zare, S. Abolfazli, M. Shojafar, and A. Kamsin, "Resource scheduling in mobile cloud computing: Taxonomy and open challenges," in *Proc. IEEE Int. Conf. Data Sci. Data Intensive Syst.*, Sydney, NSW, Australia, Dec. 2015, pp. 594–603.

[15] M. Abbasi, E. Mohammadi Pasand, and M. R. Khosravi, "Workload allocation in IoT-fog-cloud architecture using a multi-objective genetic algorithm," *J. Grid Comput.*, vol. 18, no. 1, pp. 43–56, Mar. 2020.

[16] N. Bisnik and A. Abouzeid, "Queuing network models for delay analysis of multihop wireless ad hoc networks," in *Proc. Int. Conf. Wireless Commun. Mobile Comput.*, Jul. 2006, pp. 773–778.

[17] R. B. Lenin and S. Ramaswamy, "Performance analysis of wireless sensor networks using queuing networks," *Ann. Operations Res.*, vol. 233, no. 1, pp. 237–261, Oct. 2015.

[18] X. Liang, M. Nguyen, and H. Che, "Wimpy or brawny cores: A throughput perspective," *J. Parallel Distrib. Comput.*, vol. 73, no. 10, pp. 1351–1361, Oct. 2013.

[19] V. Bhaskar, "A closed queuing network model with multiple servers for multi-threaded architecture," *Comput. Commun.*, vol. 31, no. 14, pp. 3078–3089, Sep. 2008.

[20] A. Roy, J. L. Pachuau, and A. K. Saha, "An overview of queuing delay and various delay based algorithms in networks," *Computing*, vol. 103, no. 10, pp. 2361–2399, Oct. 2021.

[21] M. Gonzalez-Martín, M. Sepulcre, R. Molina-Masegosa, and J. Gozalvez, "Analytical models of the performance of C-V2X mode 4 vehicular communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1155–1166, Feb. 2019.

[22] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.

JINGYUN WEI was born in Shaanxi, China, in 1999. She received the B.S. degree in information countermeasure technology from Xi'an Technological University, in 2021, where she is currently pursuing the M.S. degree with the School of Computer Science and Engineering. Her research interests include queuing networks and edge computing.

XIANGYANG LIANG received the B.S. degree from Nanjing University of Science and Technology, Nanjing, China, in 1996, the M.S. degree in computer science and technology from Xi'an Technological University, Xi'an, China, in 2004, and the Ph.D. degree in computer simulation from Northwestern Polytechnical University, Xi'an, in 2008. Since June 1996, he has been with the School of Computer Science and Engineering, Xi'an Technological University, where he is currently a Professor of computer science and technology. His current research interests include computer vision, big data analysis, artificial intelligence, system modeling, and distributed interaction simulation.